

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開平10-69418

(43)公開日 平成10年(1998)3月10日

(51)Int.Cl.⁶

G 0 6 F 12/00

識別記号

5 3 3

庁内整理番号

F I

G 0 6 F 12/00

技術表示箇所

5 3 3 J

審査請求 未請求 請求項の数9 O L (全 16 頁)

(21)出願番号 特願平9-167571

(22)出願日 平成9年(1997)6月24日

(31)優先権主張番号 9 6 1 1 0 6 4 9 . 9

(32)優先日 1996年7月2日

(33)優先権主張国 ドイツ (D E)

(71)出願人 390009531

インターナショナル・ビジネス・マシーンズ・コーポレイション

INTERNATIONAL BUSINESS MACHINES CORPORATION

アメリカ合衆国10504、ニューヨーク州
アーモンク (番地なし)

(72)発明者 フランク・ライマン

ドイツ連邦共和国 デー71134 アイト
リンゲン ハーゼンエッカーヴェーク 19

(74)代理人 弁理士 坂口 博 (外1名)

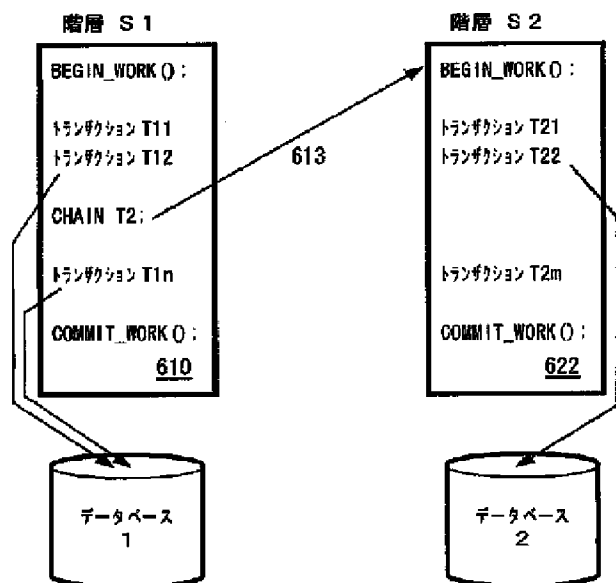
最終頁に続く

(54)【発明の名称】 階層化トランザクション処理方法

(57)【要約】

【課題】潜在的に分散した多数のトランザクションの総体のアトミック性を保証すると同時に、前記トランザクションの協調のための通信量ならびに前記トランザクションの総体の並行性挙動および処理量を最適化する。

【解決手段】本発明の根本的な概念は、トランザクションの総体をトランザクションのグループに分割する手法に基づく。各グループは、「トランザクション階層」または単に「階層」と呼ばれる。各階層は、アトミック・コミット・プロトコルによって処理され、同期化されて、それにより、個々の階層ごとにACID処理結果が保証される。したがって、得られるトランザクションのグループの総体として、すなわち階層の総体として構成される大域トランザクションは、連鎖トランザクション処理の原理にしたがって、連鎖階層のセットとして処理される。連鎖階層実行は、大域トランザクションのアトミック実行を保証する。



階層化トランザクション

ST = (T 1 1 , . . . , T 1 n , T 2 1 , . . . , T 2 m)

【特許請求の範囲】

【請求項1】アプリケーション・プログラム定義コミット有効範囲に関してアトミック・コミット・プロトコルによってアプリケーション・プログラムにACID性機能を提供する少なくとも一つのコンピュータ・システムによるトランザクション処理方法において、それぞれが少なくとも一つのメンバ・トランザクションを含む、第一のトランザクション階層および第二のトランザクション階層を少なくとも含むことと、第一のステップにおいて、コミット有効範囲をもつ前記第一のトランザクション階層を処理して、前記アトミック・コミット・プロトコルによって処理結果のACID性を保証し、さらに、前記第一のステップにおいて、前記第二のトランザクション階層の処理を要求するトランザクション実行要求を待ち行列に挿入することと、連鎖トランザクションの概念にしたがって、前記第一のステップを、前記第一のトランザクション階層がその処理を正常にコミットする場合にのみ処理される第二のステップと連鎖させることと、前記第二のステップにおいて、前記待ち行列から前記トランザクション実行要求を取り出すことを含み、さらに、前記第二のステップにおいて、コミット有効範囲をもつ前記第二のトランザクション階層を処理して、前記アトミック・コミット・プロトコルによって処理結果のACID性を保証することと、を特徴とする方法。

【請求項2】前記待ち行列への前記トランザクション実行要求の挿入が、前記第一のトランザクション階層の前記コミット有効範囲の一部であり、前記待ち行列からの前記トランザクション実行要求の取り出しが、前記第二のトランザクション階層の前記コミット有効範囲の一部である請求項1記載のトランザクション処理方法。

【請求項3】前記待ち行列が多数の異なるコンピュータ・システムに分散している、かつ／または、前記待ち行列に挿入される前記要求がトランザクション保護によって扱われる請求項1または2記載のトランザクション処理方法。

【請求項4】前記待ち行列が、前記トランザクション実行要求を非同期に処理することによって非同期トランザクション処理をサポートしている請求項1～3のいずれか1項記載のトランザクション処理方法。

【請求項5】前記待ち行列が、リソース・マネージャによって管理される耐性保護リソースを表している請求項1～4のいずれか1項記載のトランザクション処理方法。

【請求項6】前記第一のトランザクション階層および／または前記第二のトランザクション階層によって構成される前記メンバ・トランザクションまたは前記メンバ・トランザクションの一部が、何らかの種類のコンピュータ・ネットワークによって接続された多数のコンピュータ・システムによって処理される請求項1～5のいずれ

か1項記載のトランザクション処理方法。

【請求項7】前記アトミック・コミット・プロトコルが2相コミット・プロトコルまたは3相コミット・プロトコルとして実現されている請求項1～6のいずれか1項記載のトランザクション処理方法。

【請求項8】多数の異なるポリシーが、前記階層それぞれの中で処理されるメンバ・トランザクションを定義する請求項1～7のいずれか1項記載のトランザクション処理方法。

【請求項9】局所ノード・ポリシーが、同じトランザクション階層の中で処理すべき、同じコンピュータ・システム上で実行されるすべてのメンバ・トランザクションを定義する請求項8記載のトランザクション処理方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、コンピュータ化トランザクション処理方法に関する。より詳細には、本発明は、トランザクションの総体のアトミック性を保証する、潜在的に分散した多数のトランザクションの総体のコンピュータ化処理に関する。

【0002】

【従来の技術】トランザクション・アプリケーションとも呼ばれるトランザクションは、回復可能なリソースおよびデータ、たとえばデータベースを一つの整合状態から別の整合状態に変化させる一連の処理を含む。トランザクション処理システムは、あるトランザクションが回復可能なリソースまたはデータに対して何らかの更新を実行し、そのトランザクションがその正常な終了または整合性の中間点に達する前に障害が起こるならば、そのような更新が取り消されること（ロールバック）を保証する。新たな整合性の点に達し、トランザクションによって実施されたすべての更新が永久的にされなければならないとき、トランザクションはコミットする。

【0003】このトランザクション回復保証を履行するためには、システムは、システム動作不能状態の間、進行中のトランザクションおよびそれらの更新動作の両方を、システムが再起動したとき、回復可能データに対するそれらの効果を正しく反映させることができるよう、記憶することができなければならない。一般に、システムは、トランザクション回復保証を履行するために、ログ・マネージャによって記録され、維持される、各トランザクションの進行状態ならびにリソースおよびデータに対する変更に関するログを維持する。ログ・レコードとして知られるログ・データを審査すると、トランザクションのコミットした動作が、データベースに反映されているか、取り消されたかのいずれかであることを保証することができる。また、ログ・レコードが実データを含むときには、ログ・レコードを利用して、たとえば記憶装置の障害によって損傷または損失したデータを再構成することもできる。ログは、常に成長する順次アクセ

ス・ファイルと考えることができる。

【0004】ログは、システム障害の際にも無傷で利用可能な状態にとどまる安定な記憶装置、たとえばディスク・ファイル上に永久的に記憶されている。ログ・レコードは、まず、コンピュータのメモリの中の一時的な揮発性ログ・ファイル・バッファに書き込まれたのち、一定の時機（たとえば、トランザクションがコミットされたとき）に安定な記憶装置に移される。

【0005】ロック・マネージャによってサポートされる「ロッキング」は、共用リソースおよび共用データに対して同時に実行される（すなわち、並行な）トランザクションのアクセスを制御するのに使用され、特に、並行トランザクションが同じリソースおよびデータを不整合に変更してしまうことを防ぐのに使用される。ロッキングを適切に使用すると、トランザクションが操作するリソースおよびトランザクションが読み出すデータが整合状態にあり、リソースおよびデータが他のトランザクションのコミットしていない更新を含まないことをトランザクションに保証することができる。

【0006】トランザクションが操作する回復可能なリソースおよびデータはさらに、リソース・マネージャによってサポートされることもできる。リソース・マネージャとは、一定のタイプのトランザクション・オブジェクトを管理するサブシステムである。リソース・マネージャは通常、システム全体で利用可能であるかもしれないサービスを、アプリケーションまたは他のリソース・マネージャに提供する。トランザクション・データベース・システム、トランザクション待ち行列マネージャ、トランザクション・セッション・マネージャなどはすべてリソース・マネージャとして働くことができる。

【0007】トランザクション・マネージャは、コンピュータ・システムの中で並行に処理が起こる多数のトランザクションの流れを統制し、管理し、協調させる。これは、トランザクションのコミットおよび取り消し、すなわちロールバックならびにそれらが失敗したのちのオブジェクト、リソース・マネージャまたはサイトの回復を命令する。

【0008】局所または分散トランザクション・システム中での一定のトランザクションおよび他すべての並行に処理が起こるトランザクションに関する上述の整合性要件は、ACID要件としてより正確に表現される。実際、ACID性とは、以下に説明する4種の異なる部分的要件を組み合わせたものである。

【0009】・アトミック性 (Atomicity)
システム全体の状態に対するトランザクションの変更はアトミック性である。すなわち、すべてが起こるか、何も起こらないかである。これらの変更は、データベース変更、メッセージ、変換器に対する動作などをはじめとする、リソースに対するすべての変更を含む。

【0010】・整合性 (Consistency)

トランザクションは、システム状態の正確な変換である。一つのグループとして受け取られる動作は、状態に関連する整合性制約のいずれをも侵害しない。これには、トランザクションが正しいプログラムであることを要する。

【0011】・隔離 (Isolation)

トランザクションが並行に実行されるとしても、それぞれのトランザクションTにとっては、他のトランザクションがTの前またはTの後のいずれか（前後両方ではない）で実行されるように見える。

【0012】・耐性 (Durability)

ひとたびトランザクションが正常に完了し、その作業をコミットするならば、状態に対するその変更は障害に耐えて存続する。すなわち、状態変化は永久的になる。

【0013】コンピュータ科学の分野では、大域トランザクションのアトミック性を保証する機構が公知である。詳細は、たとえば、「J. Gray, A. Reute, Transaction Processing: Concepts and Techniques, Morgan Kaufmann Publishers, San Francisco (CA)」に見いだすことができる。焦点は、トランザクションの総体が整合性のある共通のトランザクション結果、すなわち整合性のシステム状態（すなわちコミットまたはアボートのいずれか）に達することを許す同期プロトコルである、いわゆるアトミック・コミット・プロトコル (ACP) にある。もっとも普及し、開発されているACPは、多数の異なる商業用システムにおいて幾度も実現され、使用されてきた、いわゆる2相コミット (2PC) プロトコルである。

【0014】その重要性から、2PCは、企業連合によって標準化された。たとえば、手続き可変要素は、X/OPENによって標準化された。X/OPENの詳細は、「X/OPEN Guide, Distributed Transaction Processing Reference Model (Version 2), X/OPEN Company Ltd., U.K. 1993」に見いだすことができる。オブジェクト指向可変要素は、オブジェクト管理グループ (OMG) によって標準化された。オブジェクト管理グループの詳細は、「Object Management Group, ObjectTransaction Services (OTS), OMG Document TC 94.8.4 (1994)」に見いだすことができる。対応するX/OPEN規格の多くの規格準拠インプリメンテーションが存在し、対応するOMG規格の多くの準拠インプリメンテーションが進行中であるか、最近リリースされたばかりである。

【0015】もう一つの公知のACPは、いわゆる3相コミット・プロトコルである。

【0016】一つのアトミック作業だけからなる標準的なトランザクションの表現力は非常に限られている。標準的なトランザクションが提供するよりも融通性のある制御手段が求められるであろう多くの状況が存在する。特に実世界の応用の場合には、大域トランザクションと

呼ばれるトランザクションの総体が協力して所望の処理目標を達成しなければならない。

【0017】そのような複雑なトランザクションのために、種々の異なるモデルが開発されている。例は、セーブポイント技術、入れたトランザクション、連鎖トランザクションなどである。種々のモデルの概説は、たとえば、「J. Gray, A. Reuter, Transaction Processing: Concepts and Techniques, Morgan Kaufmann Publishers, San Francisco (CA)」に見いだすことができる。

【0018】これらすべてのトランザクション・モデルは、局所または分散トランザクションの総体、すなわち大域トランザクションのためのアトミック性を保証するための媒体として単独に考慮されるとき、根本的な問題を抱えている。これらの問題に共通の根源は、一方でのアトミック性と他方での並行性との間で適切なバランスを見いだすことの難しさである。一方でアプリケーション全体の整合性を危険にさらさないようにすることは、大域トランザクションの終了までホールドされるロックの数を増すことを意味する。これは、結果として、アトミック性を保証する協調制御努力の途方もない増大をもたらし、それにより、特に分散処理環境において、性能劣化および並行性における深刻な損失を同時にもたらすであろう。他方、現在の技術水準によると、協調制御努力の減少ならびに性能および並行性における改善は整合性の危険を伴う。

【0019】大域トランザクションの連鎖トランザクション・モデルは、トランザクションの総体に包含されるトランザクションが最終的に成功することを保証する。この結果は、トランザクションの総体が2PC処理に包含されると仮定することなく達成され、それにより、上述の問題のいくつかを回避することができる。

【0020】トランザクション連鎖によると、一つの、すなわち第一のトランザクションをコミットして、もはや必要とされないすべてのオブジェクトを解放し、なおも必要とされる処理コンテキストを、暗黙に開始される次の、すなわち第二のトランザクションに渡すことができる。第一のトランザクションのコミットメントと、次のトランザクションの開始とは、一つのアトミック操作にいっしょに包み込まれる。これは、逆に、他のどのトランザクションも、一方のトランザクションから他方のトランザクションに渡されるコンテキスト・データを見た(または変更した)はずがないことを意味する。連鎖トランザクション・モデルの欠点は、その固有の非同期モードの処理によるものである。第一のトランザクションは、第二のトランザクションの処理の要求を開始するだけであり、第二のトランザクションがすでに始まったかどうかに関して何らかの情報を同期的に戻すこともなく、その処理の結果に関する情報を戻すこともない。連鎖トランザクションのまさにその定義により、このトランザクションは、最終的には、すべての包含されたトラ

ンザクションがそれらのサービス要求を受けるということを保証するが、保証された時間枠の中で受けるということは保証できない。したがって、作業単位を概念を提供しながらも、そのような単位はしばしば、ユーザ(エンドユーザとプログラマの両方)がこの作業単位があるべきとみなすものとは対照的である。より理解しやすい(たとえば結合最大期間を有する)作業単位の実現することはやっかいである。

【0021】

10 【発明が解決しようとする課題】本発明は、潜在的に分散した多数のトランザクションの総体を処理して、その総体のトランザクション挙動を保証する、すなわちそのアトミック性を保証する課題に基づく。アトミック処理挙動を提供することと同時に、本発明の課題は、前記トランザクションの協調のために通信量を最適化することである。さらに同時に、本発明の課題は、並行性挙動および前記トランザクションの総体の処理能力を最適化することである。

【0022】

20 【課題を解決するための手段】本発明の課題は、請求項1によって解決される。大域トランザクションに包含される潜在的に分散したトランザクションの総体を処理するための本発明の根本的な手法は、メンバ・トランザクションの総体をトランザクションのグループに分割する手法に基づく。それぞれのグループは「トランザクション階層」または単に「階層」と呼ばれる。各階層は、アトミック・コミット・プロトコルを介して処理され、同期化され、それにより、個々の階層ごとにACID処理結果が保証される。したがって、得られたトランザクションのグループの総体として、すなわち階層の総体として構成される大域トランザクションは、連鎖トランザクション処理の原理にしたがって、連鎖階層のセットとして処理される。連鎖階層実行は大域トランザクションのアトミック実行を保証する。

30 【0023】本発明によって提案される技法は、トランザクション挙動をメンバ・トランザクションの総体に提供するために本来求められる協調および同期化の努力を有意に減らす。アトミック・コミット・プロトコルによる処理集約的な協調および同期化の努力は、同じ階層に属する比較的少数のメンバ・トランザクション(大域トランザクション全体の一部である潜在的に多数のメンバ・トランザクションと比較して)についてしか求められない。そのような比較的小さなメンバ・トランザクションのグループについてだけ、ACID性を保証するための処理の負担が求められる。

40 【0024】そのうえ、大域トランザクションを多数の階層に分割する結果として、環境全体の並行性挙動および処理能力が増大する。理由は、各階層は、2PCを介して協調された、より少数のトランザクションしか含まず、したがって、階層一つあたりにホールドされるロッ

クがより少ないからである。さらには、各階層は、並行に動作中である他の階層から独立してコミットして、ロックを解除するまでに必要な時間を減らす。換言するならば、要するに、階層化トランザクション処理が粒度を下げ、したがって、並行性およびシステム処理能力を増大させるのである。

【0025】階層を形成しているメンバ・トランザクションは、それらのACID性保証結果により、作業の単位とみなすことができるため、階層手法は「作業単位」という新たな概念を実現することができる。たとえば、作業単位は、ユーザとの対話に関して定義することもできる。ユーザの要求によって開始された階層は、コミットしたのち、それによって生じた他の階層の正常な完了を待つ必要なく、ただちに返ることができる。その結果、ユーザの応答時間は一般に短縮される。

【0026】連鎖トランザクション処理の概念による階層の連鎖処理により、トランザクションの総体のアトミック性が保証され、したがって、大域トランザクションのトランザクション挙動が提供される。

【0027】さらなる利点が請求項2によって達成される。本発明のさらなる実施態様によると、第一のトランザクション階層と第二のトランザクション階層とを連鎖するための処理作業は、第一および第二のトランザクション階層の一部である。本発明によると、階層の連鎖は、実行要求を待ち行列に挿入し、それを待ち行列から取り出すことによって達成される。

【0028】階層の一部としてのこれらの実行要求の扱いは、ACID操作に参加するための扱いを許し、階層をコミットした時点で、実行要求の扱いが正常に完了することを保証する。

【0029】さらなる利点が請求項3によって達成される。本発明のこのさらなる実施態様は、非同期トランザクション処理をサポートする待ち行列を教示している。すなわち、トランザクション実行要求を待ち行列に挿入したのち、制御は、要求されたトランザクションがその処理を開始または終了することを待たずに、挿入作業に戻る。

【0030】非同期トランザクション処理は、特にユーザ対話が関連する場合に多大な応答時間を有意に減らす。

【0031】さらなる利点が請求項4によって達成される。本発明のさらなる実施態様によると、前記待ち行列は、リソース・マネージャによって管理される耐性保護リソースを表している。

【0032】保護リソースである待ち行列は、トランザクション実行要求が、ひとたび正常に待ち行列に挿入されると、システムのクラッシュなどの場合でさえ、システムによって常に記憶されることを保証する。最後に、この技術により、一つの階層のアトミック性だけでなく、大域トランザクション全体のアトミック性が保証さ

れる。

【0033】各メンバ・トランザクション、ひいては各階層を任意の巨大で複雑なコンピュータ・ネットワーク内のどこでも処理できるようにすることにより、さらなる利点が達成される。したがって、メンバ・トランザクションおよび階層が多数のコンピュータ・システムの中に分散していてもよい。

【0034】多数の異なるポリシーが、階層内で処理されるメンバ・トランザクションを定義することを許すことにより、さらなる利点が達成される。したがって、個々の階層をアセンブルするメンバ・トランザクションの選択のための完全な自由が提供される。

【0035】たとえば、あるポリシーによると、同じコンピュータ・システム上で実行されるメンバ・トランザクションを組み合わせ、個々のトランザクション階層を形成する。このようなポリシーは、潜在的に分散したトランザクションの総体を協調させるのに必要なネットワーク通信量を減らすことができる。一つの同じ物理的機械上で処理されるすべてのメンバ・トランザクションを、必要なトランザクション・マネージャのインスタンスとともに、一つのトランザクション階層の中で処理することができるため、これは、2PCに準拠するために必要なメッセージを機械境界を越えて送ることを回避させ、階層あたりの協調処理の性能を高める。

【0036】

【発明の実施の形態】本明細書の記載がトランザクションを引用しているならば、局所トランザクションまたは分散トランザクションには限定されない。通常、本明細書の記載によると、トランザクションは、そのようなタイプのいかなるものであってもよい。

【0037】導入

すでに概説したように、トランザクション・アプリケーションとも呼ばれるトランザクションは、回復可能なリソースおよびデータ、たとえばデータベースを一つの整合状態から別の整合状態に変化させる一連の処理を含む。トランザクション処理システムは、トランザクションが回復可能なリソースおよびデータに対して何らかの更新を実行し、そのトランザクションがその正常な終了または整合性のインテリム・ポイントに達する前に障害が起こるならば、そのような更新が取り消されること（ロールバック）を保証する。新たな整合性の点に達し、トランザクションによって実施されたすべての更新が永久的にされなければならないとき、トランザクションはコミットされる。

【0038】図1は、疑似コードで表記されたトランザクション・プログラムの全体構造の概要を示す。当然、トランザクション・アプリケーション・プログラムは、いかなる従来の、または特殊なプログラム言語で書くこともできる。このトランザクション・アプリケーション・プログラムは、1行目でBEGIN WORK()を呼び出すこと

により、新たなトランザクションの開始を宣言する。その後、プログラムによって実行されるすべての操作は、このトランザクションの一部になる。また、トランザクション・アプリケーション・プログラムの使用中に他のプログラムによって実行されるすべての操作（2～3行目のプログラム有効範囲の中）もまた、トランザクションの一部になる。プログラムは、図1の5行目のCOMMIT WORK()を呼び出すことにより、トランザクションがシステム状態の完全で正しい変換であることを宣言する。ひとたびトランザクションが正常にコミットすると、トランザクションの効果は耐性かつ持続性になる。トランザクション中に何かがおかしくなるならば、トランザクション・アプリケーションは、7行目でROLLBACK WORK()を呼び出すことにより、すべての操作を取り消すことができる。トランザクションの実行中に障害が出るならば、システムは、トランザクションを一方的にロールバックさせることができる。そして、BEGIN-COMMITまたはBEGIN-ROLLBACKを使用して、ACID変換をブラケットすることができる。

【0039】トランザクション・アプリケーションの基本制御の流れおよびトランザクション処理システム中の主要部品を図2に示す。図2は以下のものを表している。

【0040】・トランザクション・プログラム200

【0041】・潜在的に多数のリソース・マネージャ210の一つ。リソース・マネージャとは、一定のタイプのトランザクション・オブジェクトを管理するサブシステムである。リソース・マネージャは通常、システム全体で利用できるかもしれないサービスをアプリケーションまたは他のリソース・マネージャに提供する。

【0042】・トランザクション・マネージャ220。これは、トランザクションのコミットおよびロールバックならびに障害後のオブジェクト、リソース、リソース・マネージャまたは局所もしくは遠隔サイトの回復を命令する。

【0043】・ログ・マネージャ230。これは、障害時にすべてのオブジェクトおよびリソースの整合性のあるバージョンを再構成することができるよう、トランザクションによって実施された変更のログを記録する。

【0044】・ロック・マネージャ240。これは、オブジェクトおよびリソースに対する並行アクセスを規制して、リソース・マネージャがトランザクション隔離を提供するのに役立つ汎用機構を提供する。

【0045】・トランザクション回復機能250。これは、トランザクション・プログラムのコミット有効範囲内で障害の場合に起動される。

【0046】図2を参照すると、BEGIN WORK()201がトランザクションを開始し、それをトランザクション・マネージャに登録し(202)、固有のトランザクション識別名203を作り出す。ひとたびアプリケーション

がトランザクションを開始すると、アプリケーションは、リソース・マネージャ204および205を呼び出し始め、データベースに対して読み書きを行ったり、局所および遠隔サイトに要求を送付したり、他のタイプのリソースを操作したりすることができる。

【0047】リソース・マネージャ210がそのトランザクションに関連する最初の要求を受け取ると、それはトランザクションに加わり(211)、局所トランザクション・マネージャ220に対し、それがトランザクションのコミットメントおよびロールバック操作に参加したいことを伝える。いくつかのリソース・マネージャがトランザクションに加わることは通常のことである。これらのリソース・マネージャがトランザクションを代表して作業を実行するとき、リソース・マネージャは、オブジェクトに対して加えた変更のリストを維持する。原則として、リソース・マネージャは、オブジェクトまたはリソースの旧値および新値の両方を記録する。トランザクション処理システムは、そのような変更を記録するためのロギング・サービス212を提供する。ログ・マネージャ230は、オブジェクトおよびリソースに対するトランザクションのすべての更新の順アクセス・フェイルを効率的に実現する。当然、リソース・マネージャは、これらの更新が何であるかをログ・マネージャに伝えなければならない。

【0048】隔離を提供するため、リソース・マネージャは、トランザクションによってアクセスされるオブジェクトおよびリソースをロックする(213)。これは、他のトランザクションがこのトランザクションのコミットされていない更新を見ることを防ぎ、他のトランザクションが、このコミットされていないトランザクションによって読み書きされるデータを変更することを防ぐ。トランザクション処理システムは、他のリソース・マネージャが使用することができるロック・マネージャ240を提供する。

【0049】トランザクション200がCOMMIT WORK()206を発すると、トランザクション・マネージャは、アトミック・コミット・プロトコル、すなわち、この場合では、2相コミット・プロトコル260を実行する。まず、トランザクションに加わったすべてのリソース・マネージャに問い合わせ、トランザクションが整合性のある完全な変換であると思うかどうか尋ねる。いずれかのリソース・マネージャがNoと答えれば(262)、その場合、コミットは失敗する。すべてのリソース・マネージャがYesと答えるならば(262)、トランザクションは正しい変換であり、トランザクション・マネージャはこの事実をログに記録し、各リソース・マネージャに対し、トランザクションが完了したことを知らせる。この時点で、リソース・マネージャは、ロックを解除し、トランザクションを完了するのに必要な他の操作を実行することができる。実行中にトランザクシ

ョンが失敗するか、あるいは、2PCの相1の間にリソース・マネージャがNoと答えるならば、トランザクション・マネージャはトランザクション・ロールバックを命令する。この場合、トランザクション・マネージャは、トランザクションのログを読み、ログ・レコードごとに、そのログ・レコードを書いたリソース・マネージャを呼び出して、そのリソース・マネージャに対し、操作を取り消すよう求める。ひとたび取り消しスキヤンが完了すると、トランザクション・マネージャは、トランザクションに加わった各リソース・マネージャを呼び出し、トランザクションがアボートされたことを伝える。

【0050】トランザクション・マネージャはまた、ノードまたはサイトが障害を起こした場合にもトランザクション回復を命令する。トランザクション・マネージャは、一つのオブジェクトの障害、リソース・マネージャの障害およびサイト全体の障害のための汎用サービスを提供する。

【0051】以下、トランザクション・マネージャがシステムの回復に役立つ方法を説明する。

【0052】サイトが障害を起こしたのち、トランザクション処理(TP)システムがすべてのリソース・マネージャを再起動する。いくつかのトランザクションは障害発生時に進行中であったかもしれない。リソース・マネージャは、それらの再起動ロジックの一部としてトランザクション・マネージャに接触する。そのとき、トランザクション・マネージャは、障害が起きたときに作業中であった各トランザクションの結果をリソース・マネージャに知らせる。いくつかはコミットしていたかもしれないが、いくつかはアボートされていたかもしれない、いくつかはコミットする過程にあったかもしれない。リソース・マネージャは、そのコミットした状態を独立して回復することもできるし、トランザクション・マネージャによるログの取り消しおよび繰返しスキヤンに参加することもできる。

【0053】リソース・マネージャは障害を起こすが、TPシステムの残りが動作を続けるならば、トランザクション・マネージャは、そのリソース・マネージャに関連するすべてのトランザクションをアボートする。リソース・マネージャがサービスに戻ると、トランザクション・マネージャは、それらのトランザクションの結果をリソース・マネージャに知らせる。リソース・マネージャは、この情報およびトランザクション・ログを使用して、その状態を再構成することができる。

【0054】特定のオブジェクトは損失したが、リソース・マネージャが他の方法で動作しているならば、リソース・マネージャは、他のオブジェクトに関してサービスを提供し続け、その間、失敗したオブジェクトがアーカイブ・コピーおよびそのコピーに対してコミットしたすべての変更のログから再構成される。トランザクション・マネージャおよびログ・マネージャは、アーカイブ

コピーからのオブジェクトの回復を支援する。

【0055】各サイトは普通、別個のトランザクション・マネージャを有している。これが、各サイトが互いに独立して動作することを許して、局所自律性を提供する。トランザクションの実行がいくつかのサイトの中に分散しているとき、それは、いくつかのトランザクション・マネージャの中に分散する。その場合、多数のトランザクション・マネージャの中で2PCプロトコルが使用される。

10 【0056】リソース・マネージャによって維持、制御される、トランザクション・オブジェクトとも呼ばれるリソースは、保護リソースまたは回復可能リソースである。保護リソースに対して向けられる動作は、それらの動作が完全に終了する前にそれらの結果を外部に出すことはない。これらの更新はコミットメント制御され、正常に終了する前に何かがおかしくなるならばロールバックすることができ、ひとたび正常終了に達すると、一方向のロールバックはなくなる。このように、保護リソースに向けられる動作はACIDの性質を有する。

20 【0057】図3は、分散TPシステムの側面により焦点を当てたTPシステムを示す。さらに、図2に概略したモデルとは異なるいくつかの変更を示す。図3のモデルは、X/OPEN企業連合の観点のラインに沿っている。

【0058】図は、コンピュータ・ネットワーク303を介して接続された二つのTPシステム301および302を示す。両TPシステムは、それらの間の通信を司る特定のリソース・マネージャ、すなわち通信マネージャ304および305を含む。さらにこのモデルでは、各リソース・マネージャ306および307が専用のログおよび専用のロック・マネージャを有するものと仮定する。トランザクションがアボートする場合、リソース・マネージャは、トランザクション・マネージャからの一回のロールバック呼び出しに基づいて、それ自身のロールバックを実行すると考えられる。

【0059】大域トランザクション一つのアトミック作業だけからなる標準的なトランザクションの表現力は非常に限られている。標準的なトランザクションが提供するよりも融通性のある制御手段が求められるであろう多くの状況が存在する。特に実世界の応用の場合には、トランザクションの総体が協力して所望の処理目的を達成しなければならない。そのような伝統的なACIDトランザクションの総体は、そのアトミック性が保証されるならば、「大域トランザクション」と呼ばれる。

【0060】コンピュータ科学の分野では、大域トランザクションのアトミック性を保証する機構が公知である。大域トランザクションの問題を扱う一つの手法は、いわゆる「連鎖トランザクション(Chained Transactions)」である。連鎖トランザクションは、ACID性を保

13

証するための同期アトミック・コミット・プロトコル（ACP）に関する処理努力をある程度減らすための特別な手法を提供する。

【0061】CTが、標準的なメンバ・トランザクションT1～Tnからなる連鎖トランザクションであると仮定する。連鎖トランザクションの概念は、長く続く大域トランザクションCTに対して順副階層構造を課すことを許す。この概念は、メンバ・トランザクションTiの中から、「CHAIN」実行要求により、メンバ・トランザクションTjのトランザクション処理要求を出すことを許す。二種類の連鎖トランザクションが可能である。同期連鎖トランザクション・モデルによると、メンバ・トランザクションTiのコミットメントは、結果的に、連鎖トランザクションTjの実行を同期的にもたらし。他方、非同期連鎖トランザクション・モデルでは、メンバ・トランザクションTiのコミットメントの後には、連鎖トランザクションTjの非同期実行が続く。以下、はっきりと必要ではない限り、連鎖トランザクション原理に関するこれらの異なる表現を区別しない。

【0062】Tiの処理に拘束された潜在的な実行コンテキスト、たとえばデータベース・カーソルなどを包含するデータベース・コンテキストは、保存し、Tjの処理に渡すことができる。したがって、トランザクション連鎖を用いると、一つのトランザクションTiをコミットして、もはや必要ではないすべてのリソースおよびオブジェクトを解放し、なおも必要とされる処理コンテキストを、TP処理システムによって暗黙に開始される次のトランザクションTjに渡すことができる。第一のトランザクションTiのコミットメントと、次のトランザクションTjの開始とは、一つのアトミック操作にいっしょに包み込まれる。これは逆に、他のトランザクションが、一方のトランザクションTiから他方のトランザクションTjに渡されるコンテキスト・データを見た、または変更したはずがないことを意味する。

【0063】連鎖ステップはトランザクション（Ti）を取り消し不能に完了するため、ロールバックは、連鎖トランザクションCTの中の現在作業中のメンバ・トランザクション（Tj）に限定される。Tiのコミットは、トランザクション・アプリケーションCTが、その後は必要としないロックを解除することを許す。連鎖トランザクション方式による再起動の扱いは、最新のコミット、すなわち、もっとも最近にコミットされたメンバ・トランザクションのコミットの状態を再び確立することによって実行される。

【0064】メンバ・トランザクションTiの中で次のメンバ・トランザクションTjを連鎖するとき、この処理要求が、呼び出し側トランザクションTiのコミット有効範囲の一部として、要求待ち行列に挿入される。この要求待ち行列は当然、回復可能、すなわち耐性のリソースであり、この挿入処理はACID性の要件を満た

14

す。したがって、これらの多数の回復可能な要求待ち行列は、持続性のある回復可能なメッセージングをサポートする。次のメンバ・トランザクションTjの処理要求を待ち行列から取り出すことは、Tjのコミット有効範囲の一部である。

【0065】CTの中でメンバ・トランザクションTiから次のメンバ・トランザクションTjを連鎖する処理が本質的に非同期的であるならば、これらの非同期トランザクション処理要求は、結果として、次のメンバ・トランザクションTjの処理を同時にはもたらし。その代わりに、処理要求は、呼び出し側トランザクションTiのコミット有効範囲の一部として要求待ち行列に挿入される。この処理要求が、結果として、要求された次のメンバ・トランザクションTjの実行をいつもたらしかは、TPシステムしだいである。

【0066】図4は、メンバ・トランザクションT1、T2、T3、すなわち401、402、403を包含する連鎖トランザクションCTの簡単な例を示す図である。そのコミット有効範囲の中で、T1は、ACID性が保証される特定の作業を実行する。特定のデータベースのアクセス407は一つの例である。同じくそのコミット有効範囲の中で、すなわちその作業をコミットする前に、T1は、連鎖トランザクションCTの一部である次のメンバ・トランザクションT2の実行要求を出す（404）。ひとたびメンバ・トランザクションT2が制御を得ると、それは、同様な方法で、それが司る作業を実行する。特定の時点で、それは、連鎖トランザクションCTの一部である最後のメンバ・トランザクションT3の実行要求を出す（405）。この例によると、メンバ・トランザクションT3は、制御を得たのち、そのコミット有効範囲の中でさらなるデータベースにアクセスする（406）。

【0067】本発明はまた、トランザクションがネットワーク中に分散している場合をも取り扱う。そこで、本発明はまた、分散トランザクション処理の分野にも関連する。これは、メンバ・トランザクションそれぞれが、ホストのオペレーティング・システムの別個のスレッドまたはプロセスの中で、またはネットワーク中の異なる物理的機械上で実行するかもしれないことを意味する。したがって、概念的に、一定のトランザクションがネットワーク中のどのコンピュータ・システム上で処理されるのかを区別することはもはや必要ではない。

【0068】ネットワーク内のコンピュータ・システム間の通信目的のためのメッセージ指向性ミドルウェア（MOM）の出現により、分散トランザクションの処理のための新たな概念が可能になった。過去数年間、メッセージ指向性ミドルウェア（MOM）はその重要性を増した。MQシリーズをもって、IBM社は、現在、MQシリーズのプロトコルを主に採用するISOおよびOMGのような公の標準化機関で進行中の作業によって示さ

15

れる、この分野における事実上の基準を確立した。IBM社のMQシリーズに関しては、たとえば「MQSeries Planning Guide, Document Number GC33-1349-01」を参照されたい。メッセージ指向性ミドルウェアは、リソース・マネージャによって管理される保護リソースの意味での回復可能リソースとして扱われる持続性の回復可能待ち行列を提供する。すなわち、そのような待ち行列の操作をトランザクション内で実施して、メッセージ待ち行列を扱うためのACID性サポートを保証することができる。

【0069】図5は、MOMを利用して、連鎖トランザクション・モデルにしたがって分散大域トランザクションを処理する方法を例示する。MOMの利用は、二つの連鎖トランザクションの同じ結果を保証するため、非同期プロトコルを考慮している。トランザクション・アプリケーションが二つのメンバ・トランザクションT1(501)およびT3(503)に分割され、関連の分散アプリケーションの意味的に正しい実行を結果的にもたらしめためには、T1およびT3がいずれも起こらなければならない(すなわち、コミットしなければならない)と仮定する。この場合、T1は、それがメッセージを出して(504)、その局所的な持続性回復可能待ち行列Q1(505)の中でT3を開始したものと同じ局所トランザクションの中でその対象の回復可能リソースを操作しなければならない。Q1からQ2(506)(T3に局所的な待ち行列)へのメッセージの伝達は、利用されるメッセージング・ミドルウェア層によって保証され、したがって、第三の独立したトランザクションT2(502)として認められることができる。T3は逆に、この要求を、それがその対象の回復可能リソースを操作したものと同じ局所トランザクションの中でその局所待ち行列Q2から得なければならない。その結果、T1がその作業をコミットするならば、T3は、最終的に、その関連の作業を正常に実行する。

【0070】現在の技術水準によって処理される大域トランザクションは、潜在的に分散したトランザクションの総体のアトミック性を保証する根本的な課題を抱えている。

【0071】たとえば分散トランザクション・アプリケーションは、それぞれがネットワーク中のどこかで実行している二つ以上の意味的に関連した実行スレッドからなることができる。この場合、スレッドの概念は、そのもっとも一般的な意味で、実行中の1片のコードのOS表現として使用され、オペレーティング・システムのディスパッチまたはスワップの単位の厳密な意味では使用されない。このスレッドの総体のサブセットがトランザクションからなるとき、これらのトランザクションの結果は同期化されなければならない。普通、すべての参与するトランザクションがコミットするか、すべてがアボートされるかのいずれかであることが保証されなければ

16

ならない。2PCは、このアトミック性の課題の解決方法である反面、トランザクションの総体と同じトランザクション結果(すなわち、コミットまたはアボートのいずれか)に達することを許す同期プロトコルであるACPプロトコルのファミリーの一つの代表として、いくつか周知の欠点を抱えている。たとえば、大域トランザクションの最後まで保持されるロックによるシステム全体の中での並行性の低下や、2PCに固有の、環境全体の性能の低下を招く高いメッセージ複雑性である。

10 【0072】これらの欠点は、結果的に、多くの場合、2PCの利用を回避させるインプリメンテーションをもたらす。その結果、アプリケーション全体の整合性が危うくなる。クラッシュの場合には、多くの状況では、やっかいであり、しばしば誤りを起こしやすい手作業によって全体の整合性を再び確立しなければならない。

【0073】一見すると、連鎖トランザクション処理方式が代替となりうるかもしれない。連鎖トランザクション・モデルは、包含されたトランザクションが最終的には成功するということを保証する。これは、トランザクションの総体が2PC処理に包含されるものと仮定せずにそれを保証し、したがって、上述の問題を回避させる。

【0074】連鎖トランザクション・モデルの欠点は、非同期であることに基づく。まさにその定義により、連鎖トランザクション・モデルは、最終的には、すべての包含されたメンバ・トランザクションがそれらのサービス要求を受けることを保証するが、保証された時間枠の中で受けるということは保証できない。したがって、作業単位の概念を提供しながらも、そのような単位はしばしば、ユーザ(エンドユーザとプログラムの両方)がこの作業単位があるべきとみなすものとは対照的である。より理解しやすい(たとえば結合最大期間を有する)作業単位の概念を実現することはやっかいである。

【0075】そのうえ、連鎖トランザクション・モデルは、完全なACID機能性を連鎖トランザクションに全体では提供することができないという欠点を抱えている。ACID性は、個々のメンバ・トランザクションだけに利用できる。

【0076】階層化大域トランザクション
大域トランザクションに含まれる潜在的に分散したトランザクションの総体を処理するための本発明の根本的な概念は、以下の技術に基づく。

【0077】・トランザクションの総体をトランザクションのグループに分割する。各グループは「トランザクション階層」または単に「階層」と呼ばれる。

【0078】・各階層をACP(たとえば2PC)プロトコルによって処理し、同期化し、それにより、個々の階層ごとのACID処理結果を保証する。

【0079】・得られるトランザクションのグループの総体として、すなわち階層の総体として構成された大域

トランザクションを、連鎖トランザクションの原理にしたがって、連鎖階層のセットとして処理する。連鎖階層実行は、大域トランザクションのアトミック実行を保証する。

【0080】階層、すなわちグループに分割されたメンバ・トランザクションの総体を包含する大域トランザクションを処理する手法は、多くの実際の状況で、すべての参加するトランザクションがデフォルトでそれらの終了に同期的に達する必要があるという憶測に基づく。それでも存続することは、すべての参加するトランザクションが同じ判定、すなわちコミットまたはアボートのいずれかに達するということである。この状況では、すべての参加するトランザクションの結果が2PCを介して同期化される必要はない。そのうえ、大域トランザクションを階層の総体に分割する処理は、異なるポリシーにしたがって起こることができる。たとえば、分割方法の例は、ネットワークの同じTPシステムで処理されるトランザクションどうしを同じ階層に分類するものでもよい。この手法は、ACPを介する分散同期化努力を有意に減らす。もう一つの分割ポリシーによると、もっとも重要な回復可能リソースを実施するようなメンバ・トランザクション同士が同じ階層の中で処理される。一般に、参加するトランザクションの総体は、トランザクション結果をACPを介して同期化しなければならない（たとえば、利用者による作業単位の認識を満たすために）ような、直接協力するトランザクション同士のグループに分割される。利用者の視点からは、階層の構造は見えない。すなわち、それは単に一体のトランザクションとして認識される。同じ物理的機械上で動作する参加するトランザクションのセットまたはエンドユーザによって直接認識される作業単位を実行する参加するトランザクションのセットは当然、階層としての候補である。

【0081】階層間のトランザクション依存性は、一つの階層の中のトランザクションが一斉にコミットしたならば、他の階層もまたコミットするということである。この状況は、協力するトランザクションのグループ（すなわち階層）が協力するトランザクションの他のグループ（すなわち階層）からサービスを要求する場合に頻繁に見ることができる。各個々の階層のトランザクション結果は、2PCによって管理される。この手法は、各階層の処理に対するACID性を保証する。階層のセットそのものは、連鎖トランザクション（上記に概説したような連鎖トランザクションの意味で）の総体として管理され、したがって、「非同期トランザクション束」または「連鎖階層」と呼ばれる。非同期トランザクション束を処理するための連鎖手法はアトミック性を保証する。

【0082】非同期トランザクション束は、そのインプリメンテーションのために、持続性の回復可能メッセージングに依存する。このような持続性の回復可能待ち行列は、TPシステムの一部であるかもしれず、根底のメ

ッセージング・ミドルウェアによって促進されるかもしれない。MQシリーズは、使用することができる持続性の回復可能待ち行列の一例である。このため、非同期トランザクション束に参加する各階層は、その回復可能リソースを操作して、その適切なビジネス機能を、それが独立型のトランザクションであるかのように実行する。これに加えて、階層内の各トランザクションは、根底のメッセージング層の持続性の回復可能待ち行列にメッセージを入れて、包含する分散アプリケーションの意味的成功のために、前者のトランザクションが確かに実行されなければならない、他の階層のトランザクションからのサービスを要求することを可る。アトミック性を達成するためには、これらの要求を待ち行列に入れる動作が適切な各トランザクションの一部として起こるということが決定的である。代わって、同じ理由から、待ち行列中の要求によって開始される階層内の各トランザクションは、要求そのものを取り出すことがその適切なトランザクションの一部であることを確かめなければならない。その結果、階層の中でトランザクションによって実行されるすべての「PUT」および「GET」は、包含する階層がコミットする場合にコミットされるか、包含する階層がコミットする場合にしかコミットされない。

【0083】潜在的に分散したトランザクションの総体を階層に分類し、階層のセットを非同期トランザクションの束として構成すると、得られたトランザクション・モデルを「階層化トランザクション (Stratified Transaction)」と呼ぶ。

【0084】「PUT」および「GET」を適切なトランザクションに参加させるためには、根底の待ち行列インプリメンテーションは、それ自体がリソース・マネージャとして働かなければならない。特に、ACP（たとえば2PC）プロトコルに参加しなければならない。さらには、待ち行列インプリメンテーションの運搬機構は、ひとたび待ち行列システムが「PUT」を待ち行列にコミットすると、その送達を保証されなければならないという意味において処理されなければならない。

【0085】分散トランザクションの総体を協調させるのに要するネットワーク通信量は、2PCに参加するトランザクションを制限して、すべてを一つの同じ物理的機械上で動作させることによって減らすことができる。これは、特定の階層に関連するすべてのトランザクションが（理想的には）同じ機械上で動作するということを意味する。当然、特定の機械またはTPシステムが多数の階層のホストであってもよい。連鎖トランザクション・モデルを利用して、階層そのものを参加するトランザクションとして協調させることにより、階層化トランザクション全体の整合性のある結果が達成される。

【0086】図6は、二つのトランザクション階層S1（610）およびS2（620）に分割された（何らかのタイプのポリシーにしたがって）簡単な階層化トラン

10

20

30

40

50

ザクションSTの例を示す。階層S1は、S1のコミット有効範囲で処理され、したがって、階層S1の結果にACID性を保証するn個のメンバ・トランザクションT11(611)～T1n(612)を含む。S1のコミット有効範囲の一部としての階層レベル上の連鎖機構613を介して、S1は、階層S2の非同期処理を出す。ひとたび階層S2が制御を取得し、処理を開始すると、そのm個のメンバ・トランザクションT21(621)～T2m(622)は、S2のコミット有効範囲で処理されて、ACID性を保証する。上記の説明によると、メンバ・トランザクションT11～T1nおよびT21～T2mのグループの中でだけ、ACPプロトコルは処理結果を同期化する。

【0087】図7は、図6の場合を、ただし、個々のトランザクション階層を連鎖するのにMOMを使用する形態で示す。階層S1の中では、「PUT」701によって階層S2の処理要求を挿入することにより、さらなる階層S2が開始される。分散階層化トランザクションSTの意味的に正しい実行をもたらすためには、S1は、S1のコミット有効範囲の中でその対象の回復可能リソースを操作しなければならない。したがって、S2を開始するメッセージをその局所的な持続性の回復可能待ち行列に挿入することは、S1のコミットの一部である。Q1からQ2へのメッセージの伝達は、利用されるメッセージング・ミドルウェア層によって保証される。S2は、逆に、S2のコミット有効範囲の中でこの要求をその局所待ち行列Q2から「GET」(702)しなければならない。

【0088】最後に、図8は、本発明による、より複雑な階層化トランザクションを示す。この例の大域トランザクションは、4個のトランザクション階層S1(810)、S2(820)、S3(830)、S4(840)に分割されている。各トランザクション階層は、そのコミット有効範囲の中に、メンバ・トランザクションの総体を含む。保護メッセージ待ち行列の中にメッセージを挿入することにより、メンバ・トランザクションT11が階層S2の処理を要求し(811)、メンバ・トランザクションT12が階層S2およびS3の処理を要求し(812、813)、さらにメンバ・トランザクションT14が階層S3の処理を要求する。そのうえ、トランザクション階層S2の処理の間、メンバ・トランザクションT21が階層S4の処理を要求し(821)、メンバ・トランザクションT23が階層S4の処理を要求する(822)。

【0089】利点

トランザクションの階層化は、潜在的に分散したトランザクションの総体を協調させるのに必要なネットワーク通信量を減らすことができる。これは、種々のポリシーにしたがって、特定の階層に関して含まれるメンバ・トランザクションをグループ分けすることによって達成さ

れる。たとえば、一つのグループ分けポリシーによると、一つの同じ物理的機械上で処理されるすべてのメンバ・トランザクションを、必要なトランザクション・マネージャのインスタンスとともに、一つのトランザクション階層の中で処理することもできる。これが、2PCに準拠するために必要なメッセージを機械境界を越えて送ることを回避させ、階層あたりの協調処理の性能を高める。

【0090】大域トランザクションを多数の階層に分割する結果として、環境全体の並行性挙動および処理能力が増大する。理由は、各階層は、2PCを介して協調された、より少数のトランザクションしか含まず、各階層内のトランザクションの数の減少のため、階層一つあたりにホールドされるロックがより少ないからである。さらには、各階層は、並行に動作中である他の階層から独立してコミットして、ロックを解除するまでに必要な時間を減らす。

【0091】ユーザの応答時間は一般に短縮される。理由は、ユーザの要求によって開始された階層は、コミットしたのち、それによって生じた他の階層の正常な完了を待つ必要なく、ただちに返ることができるからである。

【0092】ネットワーク中心のコンピューティング・パラダイムおよびオブジェクト要求ブローカ・パラダイムがますます採用されると、結果的にトランザクションの拡散が起こる。これらのトランザクションは、アプリケーションに組み合わせなければならない別々のサービスを提供する。この組み合わせは、しばしば、包含されるトランザクションの協調した結果を保証しなければならない。本発明は、現在の技術水準の機構に比較して、概説した利点を有することにより、これを提供する。

【0093】したがって、意味的な整合性を危うくすることなく、上述のパラダイムの中でアプリケーションの性能を調整するように努力する、分散アプリケーションの実現者は、本教示を利用するための候補者である。

【0094】略号

2PC 2相コミット・プロトコル
ACP アトミック・コミット・プロトコル
ISO 国際標準化機構
MOM メッセージ指向性ミドルウェア
OMG オブジェクト管理グループ
TM トランザクション・マネージャ
TP トランザクション処理

【0095】まとめとして、本発明の構成に関して以下の事項を開示する。

(1) アプリケーション・プログラム定義コミット有効範囲に関してアトミック・コミット・プロトコルによってアプリケーション・プログラムにACID性機能を提供する少なくとも一つのコンピュータ・システムによるトランザクション処理方法において、それぞれが少なく

21

とも一つのメンバ・トランザクションを含む、第一のトランザクション階層および第二のトランザクション階層を少なくとも含むことと、第一のステップにおいて、コミット有効範囲をもつ前記第一のトランザクション階層を処理して、前記アトミック・コミット・プロトコルによって処理結果のACID性を保証し、さらに、前記第一のステップにおいて、前記第二のトランザクション階層の処理を要求するトランザクション実行要求を待ち行列に挿入することと、連鎖トランザクションの概念にしたがって、前記第一のステップを、前記第一のトランザクション階層がその処理を正常にコミットする場合にのみ処理される第二のステップと連鎖させることと、前記第二のステップにおいて、前記待ち行列から前記トランザクション実行要求を取り出すことを含み、さらに、前記第二のステップにおいて、コミット有効範囲をもつ前記第二のトランザクション階層を処理して、前記アトミック・コミット・プロトコルによって処理結果のACID性を保証することと、を特徴とする方法。

(2) 前記待ち行列への前記トランザクション実行要求の挿入が、前記第一のトランザクション階層の前記コミット有効範囲の一部であり、前記待ち行列からの前記トランザクション実行要求の取り出しが、前記第二のトランザクション階層の前記コミット有効範囲の一部である上記(1)記載のトランザクション処理方法。

(3) 前記待ち行列が多数の異なるコンピュータ・システムに分散している、かつ／または、前記待ち行列に挿入される前記要求がトランザクション保護によって扱われる上記(1)または(2)記載のトランザクション処理方法。

(4) 前記待ち行列が、前記トランザクション実行要求を非同期に処理することによって非同期トランザクション処理をサポートしている上記(1)～(3)のいずれか1項記載のトランザクション処理方法。

(5) 前記待ち行列が、リソース・マネージャによって管理される耐性保護リソースを表している上記(1)～(4)のいずれか1項記載のトランザクション処理方法。

(6) 前記第一のトランザクション階層および／または前記第二のトランザクション階層によって構成される前記メンバ・トランザクションまたは前記メンバ・トランザクションの一部が、何らかの種類のコンピュータ・ネットワークによって接続された多数のコンピュータ・シ

22

ステムによって処理される上記(1)～(5)のいずれか1項記載のトランザクション処理方法。

(7) 前記アトミック・コミット・プロトコルが2相コミット・プロトコルまたは3相コミット・プロトコルとして実現されている上記(1)～(6)のいずれか1項記載のトランザクション処理方法。

(8) 多数の異なるポリシーが、前記階層それぞれの中で処理されるメンバ・トランザクションを定義する上記(1)～(7)のいずれか1項記載のトランザクション処理方法。

(9) 局所ノード・ポリシーが、同じトランザクション階層の中で処理すべき、同じコンピュータ・システム上で実行されるすべてのメンバ・トランザクションを定義する上記(8)記載のトランザクション処理方法。

【図面の簡単な説明】

【図1】疑似コードで書かれたトランザクション・プログラムの全体構造を示す図である。

【図2】トランザクション・アプリケーションの基本制御の流れおよびトランザクション処理システム内の主要構成部品を示す図である。

【図3】分散トランザクション・モデルを示す図である。

【図4】連鎖トランザクションの簡単な例を示す図である。

【図5】分散大域トランザクションを連鎖トランザクション・モデルによって処理するためにMOMを利用する方法を例示する図である。

【図6】本発明の階層化トランザクションの簡単な例を示す図である。

【図7】個々のトランザクション階層を連鎖するためにMOMを使用する本発明の階層化トランザクションの簡単な例を示す図である。

【図8】多数のトランザクション階層を用いる本発明の階層化トランザクションのより複雑な例を示す図である。

【符号の説明】

210、306、307	リソース・マネージャ
220	トランザクション・マネージャ
230	ログ・マネージャ
240	ロック・マネージャ
304、305	通信マネージャ

10

20

30

40

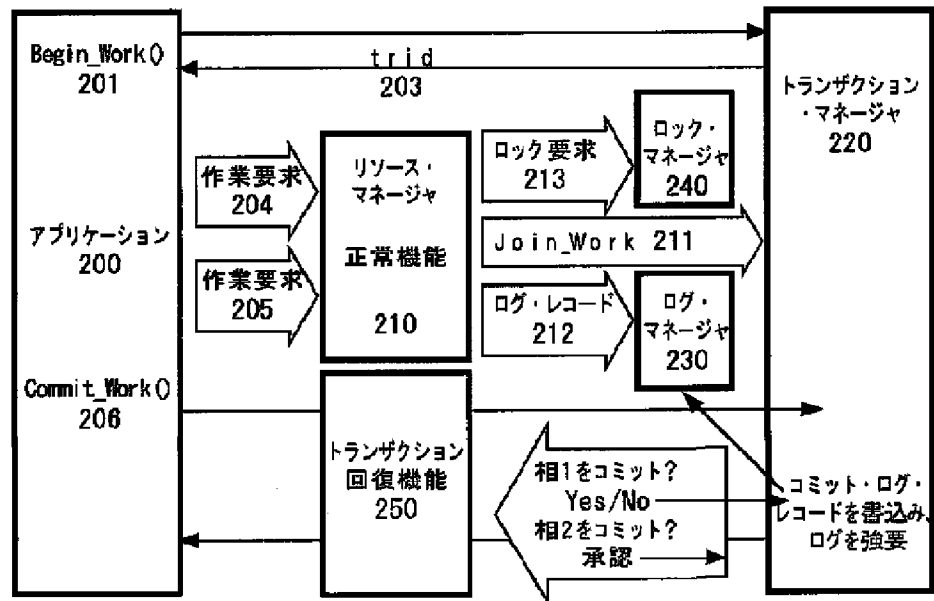
【図1】

```

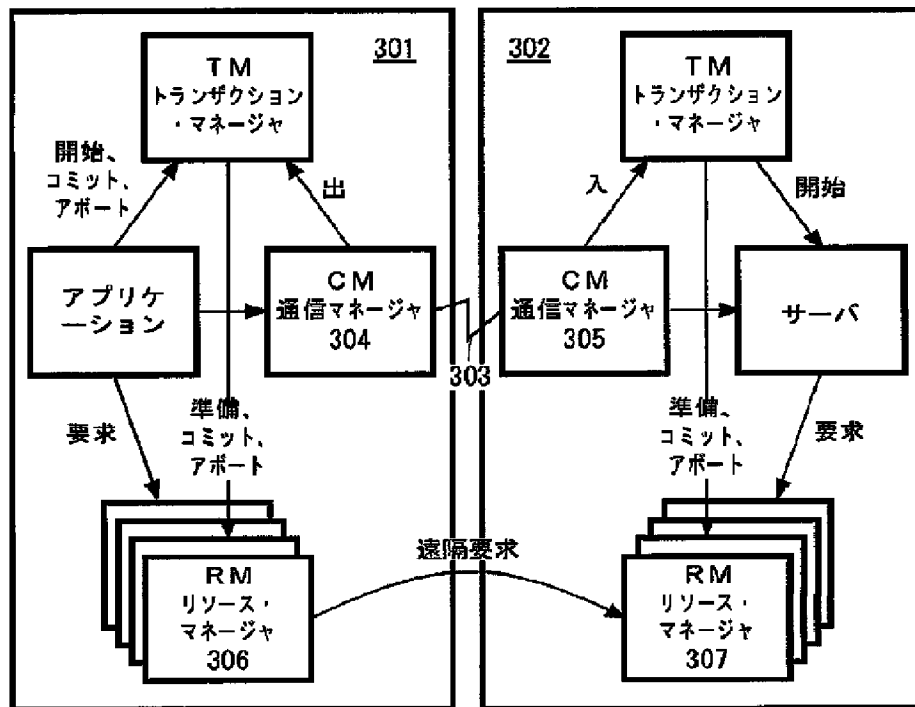
1 BEGIN_WORK();
2   <any sequence to calls to Resorce Managers>
3   <any other activity within the commit-scope>
4   if (sucesss)
5     COMMIT_WORK();
6   else
7     ROLLBACK_WORK();

```

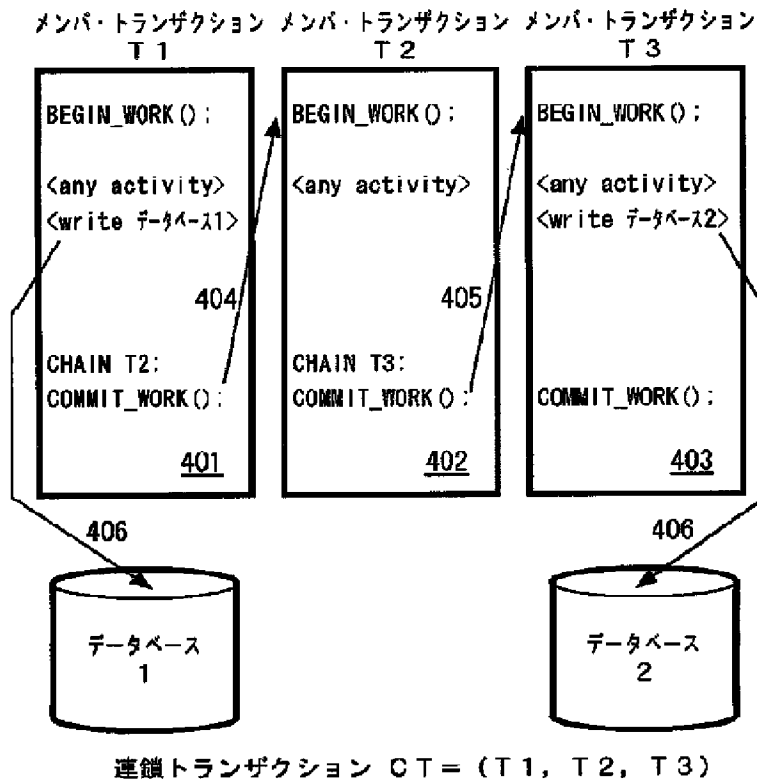
【図2】



【図3】



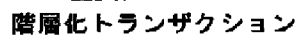
【図4】



メンパ・トランザクション メンパ・トランザクション メンパ・トランザクション
T 1 T 2 T 3

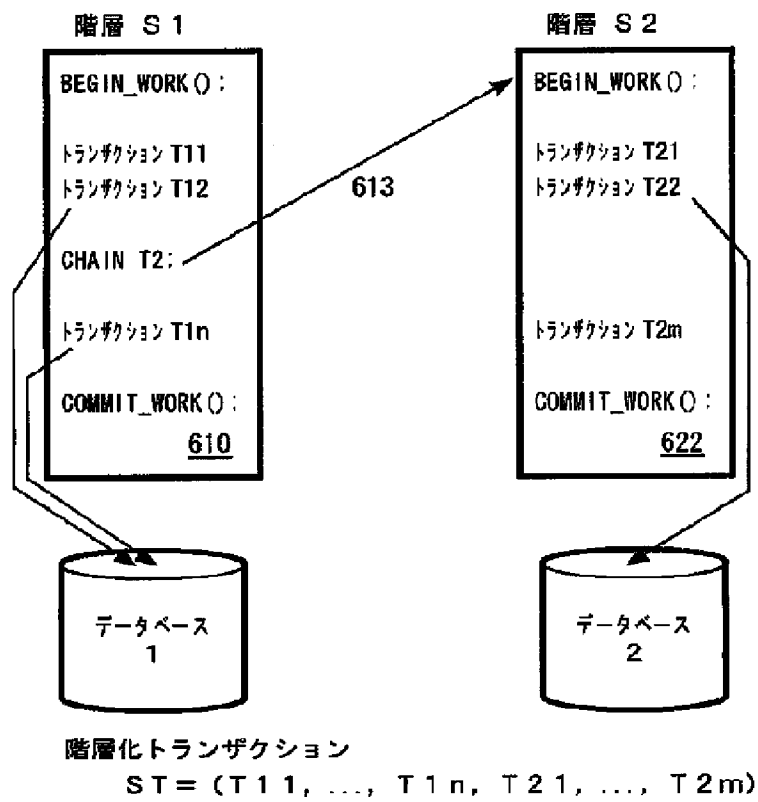


【图8】


$$ST = (T_{11}, \dots, T_{1n}, T_{21}, \dots, T_{2m})$$


=トランザクション

【図6】



フロントページの続き

(72)発明者 ディーター・ローラー
 ドイツ連邦共和国 デー71101 シェナ
 イヒヘルマン・レンス・ヴェーク 5

PAT-NO: JP410069418A
DOCUMENT-IDENTIFIER: JP 10069418 A
TITLE: HIERARCHIZED TRANSACTION
PROCESSING METHOD
PUBN-DATE: March 10, 1998

INVENTOR-INFORMATION:

NAME	COUNTRY
LEYMANN, FRANK DR	
ROLLER, DIETER	

ASSIGNEE-INFORMATION:

NAME	COUNTRY
INTERNATL BUSINESS MACH CORP	N/A

APPL-NO: JP09167571
APPL-DATE: June 24, 1997

INT-CL (IPC): G06F012/00

ABSTRACT:

PROBLEM TO BE SOLVED: To assure the total atomicity of many transactions which are latently scattered and also to optimize the traffic necessary for cooperation of those transactions and the total parallel behavior and throughput of the transactions.

SOLUTION: The fundamental concept of this

method is based on a method that divides all transactions into groups, and every group is called 'transaction hierarchy' or just 'hierarchy'. Each of both hierarchies S1 and S2 is processed by an atomic protocol and these hierarchies are synchronized with each other. Thus, the ACID processing result is assured for every hierarchy. Therefore, a global transaction consisting of all groups of transactions to be obtained, i.e., all hierarchies is processed as a set of chained hierarchies based on the chained transaction processing principle. The execution of the chained hierarchies assures the atomic execution of the global transaction.

COPYRIGHT: (C)1998,JPO